SISTEMI OPERATIVI

ESERCIZIO N. 1 del 19 DICEMBRE 2001

In un lago c'è un **servizio di noleggio imbarcazioni**, che dispone di **B barche** e **I** (< B) **istruttori**. I **turisti** che usufruiscono del noleggio si dividono in due categorie: **principianti** (che devono essere accompagnati da un istruttore) ed **esperti** (che possono navigare da soli). Una barca può essere noleggiata da un solo turista per volta, che nel caso di principiante deve essere accompagnato da un istruttore: i turisti, una volta noleggiata una barca, navigano nel lago per attraccare ad una boa, che ha capacità massima **CAP**; dopo aver ammirato il paesaggio, mollano gli ormeggi e navigano per tornare al punto di noleggio a restituire la barca. La gestione complessiva deve prevedere di dare precedenza ai turisti principianti.
Si implementi una soluzione usando il costrutto monitor per modellare il servizio di noleggio e l'attracco e i processi per modellare i **turisti** e si descriva la sincronizzazione tra i processi. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

```
program Lago

const     B = … { numero di barche }
const     I = … { numero di istruttori }
const     CAP = … { capacità del molo }
type      tipo = (princ, esperto);  { tipo di turista }

type turista = process (t: tipo)
begin
      repeat
            nol.noleggia (t);
            <naviga >
            nol.attracca (t);
            < goditi il panorama >
            nol.molla;
            <naviga >
            nol.rilascia (t);
      until false
end

type noleggio = monitor

{ variabili del monitor }
var  nmolo : integer;
      { numero di turisti sul molo }
      Boccupate : integer;
      { numero di barche occupate }
      Ioccupati : integer;
      { numero di istruttori occupati }
      coda : array[tipo] of condition;
      { code su cui sospendere i turisti in attesa della barca o
      dell'istruttore }
      codaattracco : array[tipo] of condition;
      { code su cui sospendere i turisti in attesa di attraccare }
```

```
procedure entry noleggia (t: tipo)
begin
      if t = princ
      begin
            { se non ci sono barche o istruttori }
            if Boccupate = B or Ioccupati = I then
                  coda[t].wait;
            { occupo le risorse }
            Boccupate++;
            Ioccupati++ ;
      end
      else { t = esperto }
      begin
            { se non ci sono barche }
            if Boccupate = B then
                  coda[t].wait;
            { occupo le risorse }
            Boccupate++;
      end
end

procedure entry attracca (t: tipo)
begin
      { se non c'è posto sul molo }
      if nmolo = CAP then
            codaattracco[t].wait;
      { occupo la risorsa }
      nmolo++;
end

procedure entry molla
begin
      { rilascio la risorsa }
      nmolo--;
      { se c'è un principiante in attesa }
      if codaattracco[princ].queue then
```

```
            codaattracco[princ].signal;
        else
            { risveglio un esperto }
            codaattracco[esperto].signal;

end

procedure entry rilascia (t: tipo)
begin
        if t = princ
        begin
            { rilascio le risorse }
            Boccupate--;
            loccupati-- ;
            { se c'è un principiante in coda }
            if coda[princ].queue then
                coda[princ].signal;
            else
                { altrimenti risveglio 1 esperto }
                coda[esperto].signal;
        end
        else { t = esperto }
        begin
            { rilascio le risorse }
            Boccupate--;
            { se c'è un principiante in coda
                e un istruttore libero}
            if coda[princ].queue and loccupati < I then
                coda[princ].signal;
            else
                 { altrimenti risveglio 1 esperto }
                coda[esperto].signal;
        end
end
```

```
begin { inizializzazione delle variabili }
      Boccupate := 0;
      loccupati := 0;
      nmolo := 0;
end

var  nol: noleggio; { il nostro monitor }
      p1, p2, ... : turista (princ);
      e1, e2, ... : turista (esperto);

begin end.
```

Starvation

La soluzione proposta potrebbe presentare starvation per il fatto che i turisti danno sempre la precedenza ai principianti nel risveglio.

Si può risolvere imponendo un contatore per ogni tipo di turista, alternando la priorità ogni tot di accessi consecutivi dello stesso tipo.